



# BigLittleMCA: A Spatially-Optimal Tiled Hardware Accelerator for MCMC Image Processing

CHRIS KJELLQVIST, Computer Science, Duke University, Durham, United States

LISA WILLS, Computer Science and ECE, Duke University, Durham, United States

ALVIN LEBECK, Department of Computer Science, Duke University, Durham, United States

Markov-Chain Monte-Carlo (MCMC) algorithms offer a general framework for performing interpretable inference but have high overheads due to the computational complexity of the sampling process and the large number of samples required to produce an accurate result. Computer Vision is a common class of workloads that can be performed using MCMC methods. As computer vision workloads trend toward high-resolution real-time inference, it becomes challenging to perform inference in contexts such as edge computing, which operates under strict power and area budgets. Previous work explores hardware techniques for efficient sampling; however, MCMC algorithms still require many samples. We reduce the overheads of Gibbs Sampling, an MCMC algorithm, using an approach we call mixed-resolution sampling. This approach uses low-resolution inference to provide a starting point for full-resolution sampling. We evaluate this approach on three important computer vision tasks: stereo matching, optical flow, and blind source separation. Mixed-resolution sampling reduces root mean square error (RMSE) by an average of 19.6% for stereo-matching tasks, 13% for optical flow tasks, and 6.3% for blind source separation relative to traditional Gibbs Sampling. To enable real-time, explainable MCMC inference under edge power constraints, we exploit the structure of mixed-resolution sampling to architect and implement a hardware-software co-designed accelerator architecture, BigLittleMCA (Big-Little MCMC Accelerator). BigLittleMCA is a tiled MCMC accelerator architecture that uses a small sampler for low-resolution sampling and a large sampler for full-resolution sampling. Our results show that the architecture sustains real-time 720p inference at 30 FPS (frames per second) using 48.5% less power than prior work.

CCS Concepts: • **Computer systems organization** → *Architectures*; • **Hardware** → **Application specific integrated circuits**.

## 1 Introduction

Computer Vision (CV) inference is integral to modern AI systems, such as those used in automated transportation, medical imaging, and law enforcement. To meet the growing demand for accurate CV systems, the ML community is moving towards increasingly complex, black-box models to reach high accuracy. A common shortcoming of these models is that black boxes reveal very little about why they make a certain inference. These models lack interpretability, which is especially important when AI inference may have a serious impact on human life. For example, a human surgeon uses an image segmentation model to detect a cancerous tumor for removal.

Statistical machine-learning models have broad applications in computer vision. They are an alternative to classical machine-learning methods and modern deep-learning approaches, and they provide interpretability and uncertainty measures for inferences. AI systems use uncertainty measures to describe how confident they are with an inference, which is another vital tool for systems with impactful human interactions (e.g., a vehicle's object

---

Authors' Contact Information: Chris Kjellqvist, Computer Science, Duke University, Durham, North Carolina, United States; e-mail: cmk91@duke.edu; Lisa Wills, Computer Science and ECE, Duke University, Durham, North Carolina, United States; e-mail: lisa@cs.duke.edu; Alvin Lebeck, Department of Computer Science, Duke University, Durham, North Carolina, United States; e-mail: alvy@cs.duke.edu.

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM 1544-3973/2025/5-ART

<https://doi.org/10.1145/3736171>

detection system must infer with very high confidence that a crosswalk is clear of pedestrians). While statistical models can provide accurate, interpretable results, exact inferences for practical problems are computationally intractable. Instead, most solutions use iterative Markov Chain Monte-Carlo (MCMC) methods such as Gibbs Sampling, which converge to the same result as exact inference given enough time. Unfortunately, convergence can take thousands of iterations, and each iteration requires repetitive sampling from probability distributions, which are computationally intensive due to the high overhead of sampling on conventional processors [57].

Prior work has addressed some of the computational overheads of Gibbs Sampling by using pseudo-random number generation in place of true random numbers [7, 41, 59] and by confining the problem structure to allow for highly parallel sampling. CoopMC [11], a recent work in MCMC accelerators, proposes a sampler with much higher performance-per-unit area but requires much more area for a single sampler unit. Despite these advancements, performing MCMC for increasingly larger and higher-resolution images remains expensive and requires either too many on-chip resources or too much time. Moreover, as problem sizes trend towards larger, higher-resolution images, the necessary number of iterations, and therefore the amount of work required, increases further [3, 30, 31]. *The challenge is to find solutions that go beyond per-iteration microarchitectural optimizations to reduce overall MCMC computation time, power consumption, and area.*

MCMC algorithms begin in an initially random state and iteratively sample from a distribution of possible states to produce a chain of increasingly likely states. The distribution is a function of the current iteration's state, model input, and prior knowledge. By integrating prior knowledge and past MCMC states, samples eventually converge on the distribution corresponding to exact inference. Samples collected before convergence are biased by random initialization, so they are discarded in a process called *burn-in*. Discarding results is a significant source of computational waste, but there is currently no consensus on a better method to perform burn-in. This waste creates both an incentive and an opportunity to reduce the computation time and power consumption of MCMC algorithms. We exploit this opportunity to develop a novel burn-in approach that can be efficiently accelerated in hardware.

Down-scaling image inputs is a common tactic in computer vision that has a variety of benefits, such as faster training, better accuracy, and improved robustness [13, 17, 21, 24]. The models that traditionally utilize down-scaled images are deep learning models that pool information from inference at various scales. However, it is not trivial to incorporate low-resolution inputs into full-resolution statistical sampling in the general case without additional overheads.

Since the objective of burn-in is to produce a good starting point for sampling and not to iteratively produce samples, it is not necessary to produce samples for every random variable on each iteration. We exploit this observation with low-resolution sampling: downscale the high-resolution inputs to low-resolution images and sample until convergence. We scale the low-resolution state to full resolution using nearest neighbor scaling and use this as the initial state for full-resolution sampling. We call this approach *mixed-resolution sampling*. Mixed-resolution sampling is based on a technique called *Blocking Gibbs Sampling* and shares some of the same statistical properties: it not only reduces the computational cost of burn-in but also increases the accuracy and efficiency of full-resolution sampling. We analyze mixed-resolution sampling on stereo matching, optical flow, and blind source separation applications, which are fundamental operations in CV that are used in other tasks such as scene reconstruction [51] and image compression [33]. Our results show that mixed-resolution sampling reduces end-point error relative to ground truth (i.e., sampling using full resolution, double precision floating point implementations) by 18% for stereo matching and 13.8% for optical flow.

While mixed-resolution sampling benefits software implementations, increasing their performance by approximately 2 $\times$ , we show that incorporating mixed-resolution sampling into current CPU and GPU implementations still fails to provide real-time inference (i.e., 30 FPS) on High-Definition (HD) images by one to three orders of magnitude (Figure 1). Furthermore, these approaches consume large amounts of energy to perform these inferences. We can look to hardware accelerators to address the low performance and high energy consumption.

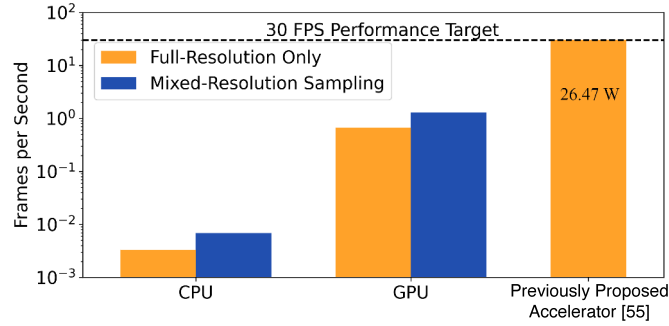


Fig. 1. Software performance in frames per second (FPS) versus prior work ASIC performance. While prior work achieves 30 FPS, it comes at the cost of high power consumption, making it unsuitable for edge deployment.

A vast body of prior work looks to accelerate MCMC-based methods using application-specific integrated circuits (ASICs) and Field Programmable Gate Arrays [7, 10, 11, 19, 38, 40, 60]. However, these works are primarily focused on improving the efficiency of the sampling process through specialized hardware samplers, often showcasing the improvements for one or a small number of samplers. While they all propose effective methods for decreasing the computational cost of the MCMC sampling operation, only a single prior work [10] explores how to tile samplers for large-scale problems. In this work, we show that these existing tiling methodologies incur unnecessary power and area overheads.

In Section 3, we propose mixed-resolution sampling and evaluate its impact on performance and accuracy for CPU and GPU architectures on stereo vision, optical flow, and blind source separation workloads. We find that while mixed-resolution sampling can decrease error rates by up to 19.6%, general-purpose computing platforms are unable to achieve real-time inference, reaching only 1.27 FPS, and use too much energy to be viable in low-power contexts. In pursuit of high-throughput inference, in Section 4, we investigate previously proposed specialized hardware architectures and find that while they can be structured to achieve high throughput, they incur unnecessary overheads when scaling to large problem sizes. We propose a novel accelerator architecture, BigLittleMCA, inspired by ARM’s big.LITTLE [1] architecture that combines mixed-resolution sampling with an existing sampler architecture to minimize these overheads and perform real-time inference of HD images at sub-15W to enable edge deployment of MCMC for computer vision tasks. Finally, in Section 5, we implement the proposed BigLittleMCA architecture and show that it performs real-time inference at 720p HD resolution using only 16.76W (48.5% less power than prior work [59]), enabling an edge-deployment scenario for MCMC accelerators performing computer vision tasks.

This paper makes the following contributions:

- We propose mixed-resolution sampling, an approximation of the Blocking Gibbs Sampling algorithm for smoothness-based priors, dramatically reducing the computational costs of burn-in sampling.
- We show that commodity CPUs and GPUs do not meet the performance or power requirements of low-power deployments, even when mixed-resolution sampling is employed, necessitating a suitable hardware design.
- We present a BigLittleMCA architecture design that supports mixed-resolution sampling to enable interpretable, real-time HD inference at the edge.

## 2 Background

In this section, we provide an overview of probabilistic inference, downscaling within computer vision, and accelerators for MCMC and outline the challenges and opportunities involved in reducing burn-in costs in MCMC algorithms.

### 2.1 Probabilistic Inference

Bayesian inference is an unsupervised learning technique that combines new evidence with prior beliefs to compute a probability distribution for a hypothesis. Performing exact Bayesian inference on probabilistic models is NP-Hard [15], making it computationally infeasible except for the smallest models. Many modern problems feature complex structures and high dimensionality, making exact inference impossible. It is common to instead use Markov Chain Monte-Carlo methods to perform inference. These methods converge to an exact solution by iteratively generating samples of the target distributions. However, these algorithms also suffer from several inefficiencies that have made them traditionally unsuitable for many applications.

While Bayesian networks can be structured arbitrarily, we discuss probabilistic inference primarily in the context of Markov Random Fields (MRFs). An MRF is a probabilistic undirected graph model in which each node represents a random variable and the edges represent mutual conditional dependence. For the image processing tasks discussed in Section 2.3, we use an MRF structured as a grid. Each pixel in an image corresponds to a random variable in the MRF, each of which is connected to its north, south, east, and west neighbors.

For this work, we focus on Gibbs Sampling, one particular MCMC algorithm. Gibbs Sampling begins by uniformly randomly initializing the state. Then, for proceeding iterations, the probability distribution for each variable given the current state is computed and sampled to update the value of the random variable for the next iteration. Prior work has largely focused on accelerating this inner loop, but this is not the only source of overhead.

MCMC state progressively becomes distinguishable from the random initialization and approaches the target distributions as iterations pass. The time it takes for an MCMC state to reach a steady state is called the *mixing time*. For large-scale problems, the mixing time can be hundreds to thousands of iterations, depending on the application and dimensionality of the problem. Since samples collected before reaching a steady state are biased towards random initialization, they are typically discarded in a process called *burn-in*.

How to best compute the mixing time is not universally agreed upon, and most practical applications rely on visual inspection of parameter trajectories or the Gelman-Rubin convergence statistic [20, 43]. Visual inspection is not an option for real-time inference, and computing the Gelman-Rubin convergence statistic is very computationally expensive. Therefore, a sufficient number of iterations, on the order of hundreds to thousands, is chosen as common practice prior to the start of inference.

From the perspective of algorithm accuracy and sampling efficiency, the only pertinent samples are those collected after burn-in. That is — *how we perform burn-in is irrelevant so long as it results in a state that is close to the target distributions*. We exploit this opportunity to perform burn-in at low resolution.

**2.1.1 Blocking Gibbs Sampling.** MCMC methods are necessary because exact computation over large Bayesian models is intractable. However, for very large models, Gibbs Sampling and other MCMC methods are susceptible to prohibitively long mixing times [45]. For this reason, prior work proposes a hybrid approach called *Blocking Gibbs Sampling* (BGS) [30, 31]. BGS breaks the model into subsets of correlated random variables called blocks. Then, BGS performs exact inference for the variables in each block. Although exact inference is intractable for full networks, blocks are small, making exact inference feasible. Blocks of size one correspond to standard Gibbs Sampling. This represents a compromise between the extremes of performing exact inference on the entire Bayesian model and sampling each variable individually. Increasing the block size decreases mixing time but also exponentially increases the computational complexity of the sampling process. Blocking represents a significant

step forward in addressing the high dimensionality of modern inference, though the algorithm’s dynamic nature makes it difficult to accelerate on hardware.

## 2.2 Computer Vision

Computer Vision (CV) is a broad class of problems that aim to infer high-level features from low-level image data. We narrow our focus to optical flow [42], stereo matching [8, 53], and blind source separation (BSS) [36, 37] problems for this work. Though our approach is applicable for any task that utilizes smoothness priors, for example, time series analysis [39], which is commonly used to model complex physical systems, and spline smoothing [54], which is utilized for nonlinear regression analysis. As well, these tasks have broad applications both by themselves and as components in CV pipelines for other tasks such as image/video compression [33], 3D scene reconstruction [51], and object tracking [35]. These applications find ubiquitous use in self-driving systems, surveillance systems, traffic monitoring, medical imaging, and numerous other industrial systems.

Stereo matching applications take two images from horizontally offset cameras as input and obtain relative depth information by computing a disparity map, which encodes the difference in horizontal coordinates between corresponding image points. Optical flow applications take two consecutive frames of video imagery as input and compute the difference in horizontal and vertical coordinates between image points in the frames. Blind Source Separation takes in different observations of the same scene, which may each have different mixtures of the same source images, and separates them into the unmixed source images. Each possible horizontal offset, 2D vector, or source pixel intensity, respectively, corresponds to a single random variable label in MCMC tasks. The task of MCMC algorithms is to find the label assignment for every random variable in the network that minimizes the network’s energy.

Performing full-resolution inference is often an expensive proposition regardless of the model. That is why schemes to perform low-granularity CV are studied extensively. Prior work uses down-scaling explicitly to reduce the computational complexity of stereo matching [14, 26] by producing an initial approximation using low-resolution inputs and only fine-tuning the result using full-resolution image data. This approach, however, is not an interpretable model. Optical flow algorithms do not typically use down-scaling and instead adopt a block-based approach [25, 28] in which one model separates the images into patches and another describes how the patches move between frames. A similar but alternative approach estimates region movement at low resolution and iteratively refines these estimates to achieve per-pixel results [44]. These works indicate that low-resolution data is sufficient and even useful for approximating results.

There are several reasons why using low-resolution data can produce a high-quality result. First, natural images usually contain mostly low-frequency data in the form of observable features. Reducing an image’s resolution tends to preserve low-frequency data, so important features are still recognizable in a low-resolution image. Another consequence of downscaling is that high-frequency data, such as noise, may also be overrepresented in the resulting image, though a common solution is to apply a blur prior to downscaling, which reduces noise in the resulting image. If image features critical to inference are preserved in the down-scaling process, the result quality will not be severely reduced.

Second, depending on the application, image inferences generally exhibit a high degree of smoothness (i.e., pixels on an object are likely to behave similarly to each other). Therefore, the results produced from low-resolution and high-resolution inference will likely be similar.

Finally, high-resolution image processing problems often require more labels per random variable. For instance, the labels for stereo vision and optical flow, 1-D and 2-D offsets, will need to encode larger offsets for higher-resolution images. As a result, a random initialization is likely to be numerically farther away from the target distributions, necessitating more iterations to reach the target distributions.

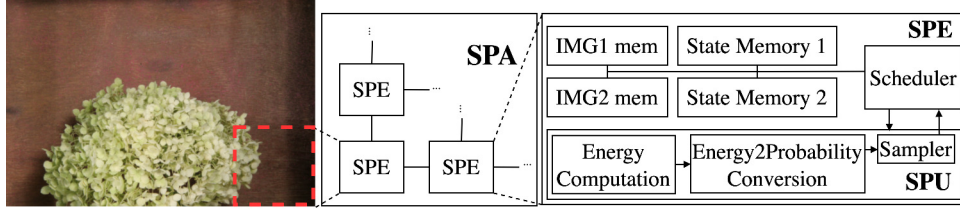


Fig. 2. Hierarchy of the SPA. Grid subdivisions of the work are assigned to each SPE, which are processed using the SPU architecture.

Prior work has explored other ways to initialize MCMC states. One such work used CNN-based object detection to bound MCMC inference to a small area of interest, reducing the computational complexity of the sampling process [47]. Similarly, another work developed a pyramidal approach to importance sampling that recursively samples areas of high interest, thereby performing less work on areas of low interest [18]. However, these approaches are unsuitable for general CV tasks because they are aimed at extremely rare event modeling. Our applications simply need a sufficient number of samples over a large image as quickly and efficiently as possible – we need high throughput over the whole input.

### 2.3 MCMC Accelerators

Our architecture is inspired by the *Stochastic Processing Accelerator* (SPA) of Bashizade [10] and Zhang et al. [59, 60], which accelerates large-scale image segmentation, optical flow, and stereo matching tasks. The SPA achieves high throughput by subdividing the workload amongst a grid topology of intercommunicating *Stochastic Processing Elements* (SPEs). Each SPE contains a *Stochastic Processing Unit* (SPU), the functional sampling unit for this architecture. The SPU [59] performs high-throughput, low-power inference by utilizing various approximation techniques for sampling in hardware using integer arithmetic. The SPU comes in two flavors: fully digital computation and true random number generation through the use of analog components. We use the digital version for this work. The architecture is shown in Figure 2.

The SPA explicitly supports three primary applications, though its primitives are extensible to other applications. These applications are image segmentation, stereo matching, and optical flow.

$$E_{i,j}(l, N) = 2^\alpha \cdot \|\text{img2}_{i-l,j} - \text{img1}_{i,j}\|_2^2 + 2^\beta \sum_n^N \|l - n\|_2^2 \quad (1)$$

The SPU is a Gibbs sampling functional unit that utilizes approximation techniques and pseudo-random number generation [59, 60]. The micro-architecture comprises three stages (shown in Figure 2): energy computation, energy to probability conversion, and sampling. Equation 1 shows how the energy of label  $l$  is computed for a pixel  $\text{img1}_{i,j}$  given the state of neighboring random variables,  $N$ . The equation comprises a first term that computes the similarity between the  $\text{img1}$  pixel and the pixel in  $\text{img2}$  corresponding to label  $l$  and a second term that computes the similarity between  $l$  and the states of neighboring labels. These are considered "smoothness priors" because they are minimized when the values compared in each term are similar. Mixed-resolution sampling takes advantage of computer vision inference tasks that use smoothness priors to reduce the problem's dimensionality while maintaining the problem structure and sampling integrity. The parameters  $\alpha$  and  $\beta$  are tunable hyperparameters that dictate the relative importance of the two smoothness terms in Equation 1. We perform a design space exploration over these parameters in Section 3.1.

After the energy for each possible label is computed for a pixel, the set of energies is used to compute a probability distribution, which is then sampled, corresponding to the "Energy2Probability Conversion" and "sampler" units shown in Figure 2. These latter two stages are shared across Gibbs Sampling implementations, but the procedure for computing a label's energy differs for each application. This three-step process results in the production of a single sample for a single pixel. One iteration of the algorithm corresponds to producing a sample for every pixel, and the algorithm is run for hundreds of iterations.

A single SPE only works on a tile of the random variables in the image at a time. Multiple SPEs are conjoined in a grid to produce inference for an entire image. Equation 1 shows that a random variable depends on the state of its neighbors, which will require SPEs to communicate with each other when the random variable neighborhood falls across a tile boundary. This raises an issue: if an SPE needs to read a random variable for its own computation and its SPE neighbor on the same cycle, the random variable states must be stored in a dual-ported memory. Prior work [10] avoids this complication by having all SPEs operate in lockstep; if one SPE is trying to read a random variable state from its neighbor, then all SPEs are, and the accesses can be performed with single-ported memories, significantly reducing the area and per-access energy costs of memory cells. This design choice has ramifications that inform our later designs, namely that the tile sizes assigned to each SPE must be identical to ensure that state machines across SPEs stay in lockstep.

### 3 Mixed-Resolution Sampling

In this section, we propose and evaluate mixed resolution sampling from an algorithmic and software perspective: how it affects inference accuracy, how it interacts with algorithm hyperparameters, and its performance in software.

When performing Blocking Gibbs Sampling, blocks are formed from variables that are thought to be highly correlated. When using smoothness priors, though, distributions for correlated variables will also likely share similar medians. For this reason, we use a single random variable to represent blocks with the idea that the median of the single random variable is likely to be close to the true median of the random variables it represents. This produces a lower-dimensional problem and eliminates the need to compute the joint distribution analytically for each random variable in the block. We downscale the image inputs to match the dimensions of the new problem. Our results show that performing Gibbs Sampling on this smaller problem has a much shorter mixing time but converges to a higher error rate due to information loss in the input image.

To combat the higher error rate of down-scaled inference, we develop a novel mixed-resolution approach. We begin execution with low-resolution sampling and then scale up the resulting state to full resolution using nearest-neighbor scaling. Nearest-neighbor scaling assigns to each pixel the closest pixel in the down-scaled image. If a pixel in the full-resolution image is between two pixels in the down-scaled image, no interpolation is applied, and the closest pixel is chosen. This scaling technique produces less smoothness but maintains the hardness of edges (for example, an outline of an object in an image is preserved). The scaled-up state then serves as the starting point for full-resolution sampling. A small number of iterations are performed at full resolution, after which we end the burn-in period and begin collecting samples. Since the algorithm consists of two distinct phases, it is feasible to use different hyperparameters during each phase to acquire better results. We investigate these parameterizations in Section 3.1.

Our approach can have several consequences on inference accuracy. On the one hand, it may increase accuracy by forcing the end result to exhibit more smoothness, which aligns with our prior beliefs. On the other hand, using a discrete sampler results in much lower bit-precision when sampling a low-resolution image. Consider an image that is scaled down by a factor of  $10\times$ . When performing sampling on this image, the sampler cannot encode an offset of 0.1 pixels and will form a steady state that is unstable at full resolution. After scaling up the resulting low-resolution state, the sampler will require extra iterations to find a new steady state.

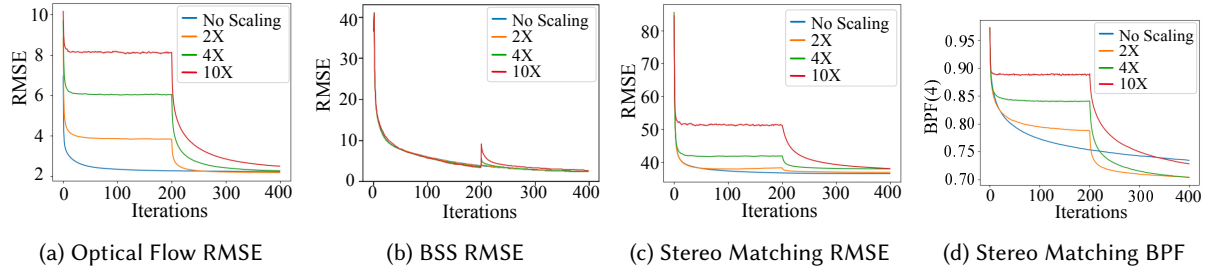


Fig. 3. Accuracy metrics for optical flow and stereo matching sampling tasks that sample for 200 iterations at low resolution and then 200 iterations at full resolution.

Finally, the particular scaling algorithm can over-represent high-frequency noise and under-represent small features in our down-scaled image. A common approach is to apply a blur to the input images prior to scaling, removing noise from the image. However, this removes information from the image and produces a gradient between features that do not exist in the input image. For instance, consider an image with only black and white pixels. Applying a blur will produce gray pixels that do not exist in the input image. Ultimately, we find that the chosen scaling approach has little effect on end-point accuracy because we perform a few full-resolution burn-in iterations after the initial down-scaled burn-in. This re-integrates any lost information and negates most of the information loss that occurs when performing inference on down-scaled images. Accordingly, we select a simple scaling algorithm and naively sub-sample without a blur to produce low-resolution inputs.

### 3.1 Algorithm Evaluation

**3.1.1 Downscaling Evaluation.** We evaluate our algorithm by first finding suitable down-sampling factors for each of our chosen applications: optical flow [42], stereo matching [8], and blind source separation [36]. We train and evaluate our approach using the Middlebury stereo vision and optical flow data sets [6, 53]. To our knowledge, there are no public high-resolution datasets for blind source separation, and it is instead typical to use synthetic mixes of generated or natural images to evaluate these methods [36]. Accordingly, we acquired a series of high-resolution grayscale photos and synthetically mixed them. We were able to reproduce the accuracy results of the original paper using these inputs. Each BSS input is given as two observed images, each a mixture of our ground truths:  $(1 * img_1 + 0.6 * img_2)$  and  $(0.6 * img_1 + 1 * img_2)$ .

Although the low-resolution and full-resolution phases can use different hyperparameters, we keep them constant in this initial analysis to isolate the effects of the down-scaling factor. Since optical flow and stereo matching infer displacements between pixels in pairs of images, the inferred displacements are also scaled down. This allows the low-resolution sampling phase to use proportionally fewer labels, reducing the work performed per iteration.

For stereo vision and optical flow, which use the smoothness function from Equation 1, we select values for  $\alpha$ ,  $\beta$ , and temperature ( $T$ ) from the full-resolution parameter tuning we perform later in this evaluation. Temperature is kept static between iterations. Blind Source Separation uses a different smoothness function parameterized by  $\delta$  and  $\beta$ . We execute 200 iterations of low-resolution sampling and 200 iterations at full resolution.

We use two metrics to estimate our error at each iteration: root mean square error (RMSE) and bad pixel fraction (BPF). RMSE measures the average Euclidean distance between the expected ground truth values and the observed labels of the random variables. BPF measures the fraction of pixels within a threshold of their expected values. We use a threshold of 4 for stereo-matching experiments and 1 for optical flow experiments. These thresholds differ because of the magnitudes of labels in each application.

Error in MCMC contexts is usually computed by collecting a distribution of samples over many iterations and reporting the error between the mean label and the ground truth. For this analysis, we record the instantaneous error at each iteration and report the geometric mean error across datasets because this better communicates the instantaneous change in information as a result of introducing full-resolution inputs. Figure 3 shows the results of this analysis.

For optical flow, we find that  $2\times$  scaling allows us to achieve lower RMSE and BPF error after 300 iterations, whereas other scaling factors never outperform the no-scaling approach. For stereo matching, more scaling factors can achieve suitable results. While no-scaling achieves the lowest RMSE across scaling factors for stereo matching, the mixed-resolution approach significantly lowers BPF error for all down-scaling factors tested.

The error rate for all applications plateaus twice: once during low-resolution sampling and again after full-resolution inputs are used. As a result, practical implementations of mixed-resolution sampling will use a small number of full-resolution samples at the end of low-resolution sampling to produce a fully burnt-in state close to a converged state.

Figure 3 shows that mixed-resolution sampling can reduce the error rate in MCMC. In the following section, we perform full parameter tuning and show that mixed-resolution sampling achieves significantly lower error for stereo matching and optical flow with many fewer full-resolution samples.

**3.1.2 Hyperparameter Tuning.** To determine the hyperparameters that produce the highest-quality results for standard and mixed-resolution sampling, we perform two-fold cross-validation using the Middlebury stereo vision and optical flow datasets. For the BSS data sets, we measure RMSE for two parameterizations of the dataset. The first is purely a mixture of the input images. The second is a mixture with added noise ( $\sigma^2 = 5.3$ ). This is a typical practice for BSS evaluations, and without added noise, many parameterizations converge perfectly to the ground truth.

We perform the tuning for two precisions: double-precision floating point (FP64) and SPU. The SPU [59] introduced several approximation techniques that affect inference accuracy, so we perform hyperparameter tuning separately using these approximations. For full-resolution-only runs, we perform 200 iterations of burn-in and 200 sample collection iterations. Our previous analysis showed that a small number of full-resolution iterations are necessary for burn-in to achieve a good result. As such, for mixed-resolution runs, burn-in consists of 180 low-resolution iterations and 20 full-resolution iterations, followed by 200 iterations of sample collection at full resolution.

For this evaluation, we measure the geometric mean of our error metrics and the Effective Sample Size (ESS) across datasets. ESS is the expected fraction of independent samples produced by a chain. A sampler that produces more independent samples from the same number of total samples is considered more efficient. We compute a modified variant of ESS for discrete samplers as described in Zhang et. al [60]. Temperature is kept static between iterations. We show the results of this parameter tuning in Table 1.

The FP64 results indicate that mixed-resolution sampling is useful in software implementations. Mixed-resolution sampling reduces RMSE by 18% in stereo matching and by 13.8% in optical flow, and reduces BPF by 18.4% in stereo matching and by 25% in optical flow. Although ESS is lower for stereo matching using FP64 precision, the accuracy is much better, and ESS is higher for optical flow.

For BSS, our error metrics are better than those that do not use mixed-resolution sampling. In fact, our  $RMSE_{\sigma^2=0}$  converged to 0, indicating that the inference converged to the ground truth. As for the BSS ESS result, we observed that many of the pixels in the produced inference converge strongly to a single discrete pixel value. This results in extremely low variance from iteration to iteration, which poses a problem for measuring the ESS fraction and pollutes the results with degenerate cases. This is a well-understood problem with measuring ESS, specifically for highly converged, discrete random variables [60]. The ESS measured in this case is neither representative nor informative.

App.	Prec.	Mode	$\alpha$	$\beta$	T	RMSE	BPF	ESS	
Stereo Matching	FP64	full only	5	4	2.5	36.74	0.65	0.47	
		mix	low	4	7	2	30.12 (↓ 17.9%)	0.53 (↓ 18.4%)	0.36 (↓ 23.4%)
			full	6	7	2			
	SPU	full only	5	4	2.5	38.26	0.63	0.51	
		mix	low	4	5	2.5	30.78 (↓ 19.5%)	0.52 (↓ 17.4%)	0.57 (↑ 11.7%)
			full	7	6	2.5			
Optical Flow	FP64	full only	4	5	2.5	1.38	0.20	0.61	
		mix	low	4	7	2.5	1.19 (↓ 13.7%)	0.15 (↓ 25%)	0.75 (↑ 22.9%)
			full	4	7	2.5			
	SPU	full only	4	4	2	1.85	0.30	0.59	
		mix	low	4	4	1	1.64 (↓ 11.4%)	0.24 (↓ 20%)	0.64 (↑ 8.4%)
			full	4	4	2.5			
App.	Prec.	Mode	$\delta$	$\beta$		RMSE $_{\sigma^2=0}$	RMSE $_{\sigma^2=5.3}$	ESS	
BSS	FP64	full only	0.5	1024		1.6	3.92	N/A	
		mix	low	0.5	32		0	3.67 (↓ 6.3%)	N/A
			full	0.1	16				

Table 1. Results of Parameter Tuning and Evaluation - We present RMSE (Root Mean Square Error, lower is better), BPF (Bad Pixel Fraction, lower is better), and ESS (Effective Sample Size, higher is better) to show the effectiveness of our approach, as well as percent improvement over each precision’s baseline. We evaluated BSS on both noiseless and noisy inputs. We report RMSE separately for these inputs as  $RMSE_{\sigma^2=0}$  and  $RMSE_{\sigma^2=5.3}$ , respectively. Red text indicates that the result is worse than the baseline.

Platform	FPS	kJ/Frame
CPU	0.0033	6.74
CPU w/ Mix-Res	0.0068	3.34
GPU	0.665	0.071
GPU w/ Mix-Res	1.278	0.032

Table 2. Software perf. and energy results for SW implementations of traditional and mixed-resolution sampling.

Using SPU approximation techniques, mixed-resolution sampling reduces RMSE error by 19.6% for stereo matching and 13% for optical flow and reduces BPF error by 17.7% for stereo matching and 20% for optical flow relative to standard Gibbs sampling on the SPU. At the same time, sampling efficiency (ESS) increases by 11.7% for stereo matching and increases by 8.4% for optical flow. In addition to reducing the overheads of the burn-in phase, *mixed-resolution sampling produces on average 10.1% more independent samples with the same number of iterations compared to the standard Gibbs Sampling approach.*

The SPU approximation techniques work well with optical flow and stereo vision applications, partly because the random variables are pixel offsets and are naturally quantized to integers. BSS infers source images from a series of mixed images, which would mimic this affinity for quantization. However, BSS infers two additional series of random variables. First is the mixing matrix for the source images, which is the ratios of the source images added together to form each observed image. Second is the noise variable, which is similar in purpose to

the temperature variable,  $T$ , used in stereo vision and optical flow and is computed on each iteration. The noise variable cannot be kept constant or scheduled like it is for stereo vision/optical flow without inducing errors. We found that quantizing the mixing matrix and noise random variables prevents convergence to a correct inference. While these variables must be sampled with high precision, there are few of them (10s of variables) compared to the image inferences (millions of variables). We present the FP64 results in Table 1 and discuss the architectural modifications necessary to support BSS in hardware in Section 4.3. For simplicity but without loss of generality, the proposed hardware implementation leverages the existing SPU implementation, which currently supports optical flow, stereo vision, and image segmentation workloads.

We conclude that mixed-resolution sampling can produce higher quality results in software and using hardware approximation techniques. In addition, mixed-resolution sampling requires less work during burn-in (8× less for optical flow, 1000× less for stereo matching, and 4× less). Unfortunately, existing hardware accelerators do not support the data movement or computation primitives necessary for mixed-resolution sampling and contain inefficiencies that increase their power usage.

### 3.2 Software Evaluation

We evaluate the performance of mixed-resolution sampling on CPU and GPU platforms with a C++ and CUDA implementation using an Intel Xeon Gold 6140M CPU with 72 cores running at 2.3GHz and an NVIDIA A5000 GPU. The CPU implementation uses double-precision IEEE floating point, and the GPU implementation uses half-precision CUDA floating point to increase memory density and performance. Full-resolution runs use 400 iterations per frame, and mixed-resolution frames use 200 iterations each of low-resolution followed by full-resolution sampling. Stereo matching uses 256 labels and 10× down-scaling when performing mixed-resolution sampling, and optical flow uses 225 labels and 2× scaling. Table 2 shows the results of this evaluation.

In grid MRFs, a "chromatic" update schedule is used to update random variables in a checkerboard pattern. The "red" random variables are updated in parallel, and only after that are the "black" random variables updated. One execution of our GPU kernel performs one chromatic update, requiring two executions to update the entire state.

GPU performance is low due to low compute utilization. GPUs achieve high compute utilization when the algorithm can be broken into many blocks, each with many threads. The Gibbs Sampling algorithm structure is three sequential map-reduce operations for a set of operands with an awkwardly large number of operands (labels); there are enough operands so that register-file pressure/shared-memory over-utilization can become issues, but not enough operands so that it makes sense to write them out to main memory (main-memory latency would otherwise become an issue). Due to the large number of intermediates per thread, the maximum number of threads per block was limited, limiting GPU utilization. We found the highest-performance GPU implementation stored operands in shared memory. We further reduce memory pressure by using NVIDIA half-precision floating-point intrinsics. While this doubles the sampling throughput of our GPU platform, an orders-of-magnitude improvement is still necessary to achieve real-time, low-power MCMC inference.

## 4 BigLittleMCA Architecture

Section 3 shows that using down-scaled inputs for burn-in can reduce computation and yield high-accuracy results, but also that re-incorporation of full-resolution data is vital to achieving the best accuracy. Current MCMC architectures do not support using separate images for burn-in and sampling. We propose a BigLittleMCA architecture for down-scaled burn-in that uses a small, power-efficient sampler for low-resolution sampling and a large, high-throughput sampler for full-resolution sampling. We use the Stochastic Processing Accelerator (SPA)[10] as inspiration for our sampling architecture, improving it to scale to larger problem sizes, exploring a spectrum of topology to reduce power consumption significantly, and designing an architecture that facilitates performing burn-in with low-resolution inputs.

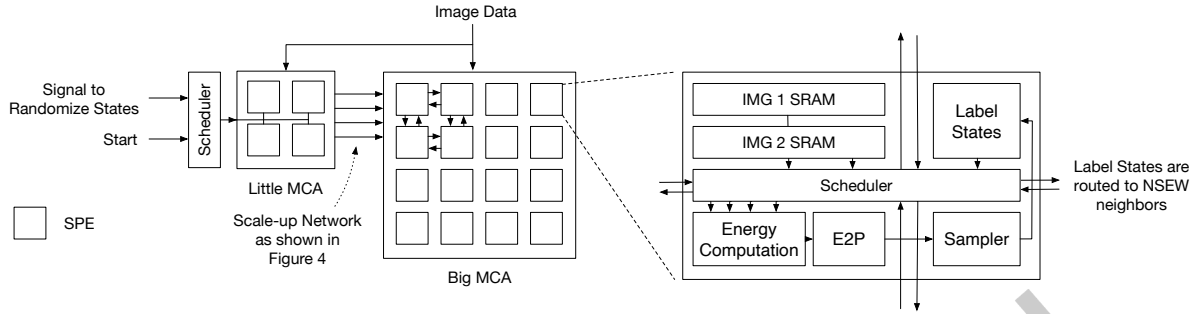


Fig. 4. Microarchitecture for the BigLittleMCA. Each MCA is composed of a grid of SPEs. Each SPE communicates label states directly with its north, south, east, and west neighbors as discussed in Section 2.3. Configuration (e.g.,  $\alpha$  and  $\beta$  parameters, number of iterations, etc.) are set once on startup and never changed (omitted from diagram for simplicity). Image data is fed in as a stream to each MCA and distributed to each SPE. The only control signals are for beginning the sampling process and signaling SPEs to randomize their internal states.

The SPU [59] sampler does not currently support some of the functions and dataflows necessary for blind source separation. As a result, we implement BigLittleMCA to initially support stereo vision and optical flow applications and discuss the necessary modifications and the associated overheads to support BSS in Section 4.3.

#### 4.1 BigLittleMCA Design

To take advantage of the power-savings of low-resolution sampling, we construct an architecture consisting of a Big and a Little augmented SPA accelerators we call MCAs (MCMC Accelerators) as shown in Figure 4. The Little MCA performs low-resolution sampling, which uses fewer labels and smaller inputs. While smaller input sizes have an obvious effect on on-chip memory requirements, using fewer labels also dramatically changes the necessary amount of on-chip memory. We explore this effect in detail in Section 4.2.

The Little MCA operates independently of the Big MCA, which performs full-resolution sampling. This allows burn-in for one frame and full-resolution sampling for another frame to execute in parallel. When the Little MCA has completed sampling, the low-resolution state is scaled up and broadcast to the SPEs in the Big MCA. One could overlay the Little MCA onto a larger MCA and perform down-scaled sampling on a subset of processing elements in a single Big MCA. While this decreases the number of processing elements on the chip, it necessitates a general-purpose NoC to distribute data during scale-up, significantly increasing the architecture complexity. This work takes the approach of using disjoint Little and Big MCAs, though many of our results generalize to an overlay approach. In this section, we consider the challenges of designing an efficient data distribution network and discuss the topology of the Big and Little MCA.

The distribution network scales the results of the Little MCA to the resolution of the Big MCA using some form of interconnect and a scheduler, some examples of which are shown in Figure 5. Since one variable in the Little MCA corresponds to many in the Big MCA, it takes multiple cycles to perform a single variable’s scale-up operation. Given a variable in the Little MCA, the scheduler is responsible for computing the destination variables that it corresponds to, determining which SPEs those variables reside on, and dispatching write requests to those SPEs.

The results from Section 3.1 indicate that different scaling factors are necessary to support both optical flow and stereo matching using the same architecture. In the most general case, the network would be able to efficiently support a software-defined scale factor and arbitrary topologies in both the Little and Big MCA. Permitting software-defined scale factors requires the use of a dynamic scheduling system. The simplest solution serially

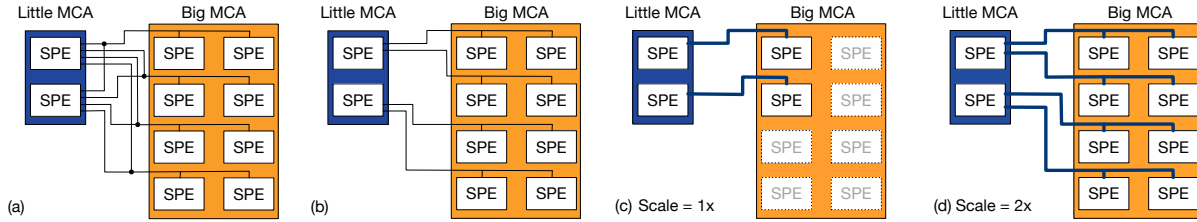


Fig. 5. Supporting all-to-all connectivity without a multi-hop network requires an expensive, high-degree crossbar network (shown in 5a). Restricting scale factors and MCA topologies results in a simpler, lower-degree network(5b). 5(c,d) show how one BigLittleMCA system supports 1 $\times$  and 2 $\times$  scale-factors. Networks for each are constructed, but only one is active at run-time. When 1 $\times$  scaling is desired, unused SPEs are power-gated. Our BigLittleMCA design uses 2 $\times$  and 10 $\times$  scaling.

dispatches random variables and only has one in-flight transmission at a time. This solution ensures that each SPE is only writing one variable at a time but has very poor performance.

Another solution dispatches random variables in parallel while ensuring that they are only dispatched to SPEs that are not currently busy processing another dispatch. Figure 6b shows how variables from different SPEs may scale up to variables residing in the same SPE in the Big MCA. A general solution that supports parallel random variable dispatches requires expensive all-to-all communication between SPEs in the Little MCA.

Without any restrictions on scaling factors or MCA topologies, the network implementation needs to support all-to-all communication from SPEs in the Little MCA to SPEs in the Big MCA. All-to-all interconnects can be structured in various ways, but implementations will either be one-hop or multi-hop networks. A single-hop global bus network implementation (shown in Figure 5a), while simple, would suffer from prohibitively large power and area penalties since MCAs can contain hundreds of SPEs. The network could alternatively be implemented as a multi-hop network, such as a torus, where packets are routed through the network to the desired SPE [32]. While the multi-hop network has acceptable resource overheads, achieving high throughput requires more complex scheduling and routing logic. Instead, we avoid the complexities of full network-on-chip architectures by restricting the generality of our scale-up solution and the MCA topologies to use statically-scheduled single-hop networks. Other network designs may be possible but will not significantly change the substantial benefits of the proposed BigLittleMCA architecture.

**Scale Factor Restriction** Section 3.1 shows that, for the data sets that we consider, only a single integer scale factor is necessary for optical-flow applications (2 $\times$ ) and that stereo vision appears to perform comparably well for a range of factors. We choose to support a small, fixed number of integer scale factors that support both applications (2 $\times$  and 10 $\times$ ), allowing us to use static scale-up schedules instead of the aforementioned dynamic scaling procedures. Without constraining the MCA topology further, the BigLittleMCA still requires all-to-all communication between SPEs in the Little MCA to prevent simultaneous dispatches to the same SPE.

**Topology Configuration Restriction** For shorthand, let Little SPEs be those that reside in the Little MCA, and Big SPEs be those that reside in the Big MCA. First, consider the case where an image can not be divided into identically sized segments amongst the SPEs in an MCA. This results in SPEs with more work than others, breaking the symmetry that allows SPEs to execute without any synchronization mechanisms [10]. Adding the necessary communication and synchronization mechanisms to permit this behavior would significantly complicate the design and add hardware overheads while not clearly contributing any performance or algorithmic benefit. Therefore, we only consider topologies that evenly divide the input resolution.

Second, consider Figure 6a,b, where an SPE in the Little MCA may issue write requests to multiple SPEs for a single variable (a) or a single SPE in the Big MCA may receive multiple write requests from several SPEs at the same time (b). In both situations, SPEs may be scheduled to perform more than one write per cycle. While this

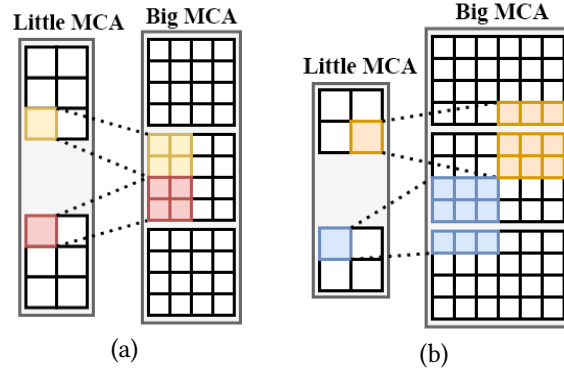


Fig. 6. 6a depicts how a 6x2 image input is distributed over a 2x1 Little MCA and then scaled up 2X to a 3x1 Big MCA. 6b shows an 4x2 scaled up 3X onto a 3x1 MCA.

behavior can be avoided using some form of dynamic scheduling, it can also be solved by adding multiple write ports to the label memory banks in the Big MCA. Both of these solutions add significant complexity and hardware overheads to the design. Instead, we avoid this by ensuring that the dimensions of the Big MCA topology are integer multiples of the corresponding dimension in the Little MCA topology.

With these restrictions, a Little SPE scales up to a disjoint set of Big SPEs. For any given scale factor, we construct an independent interconnect between Little SPE and the corresponding Big SPEs as shown in Figure 5b. Independence means dispatches can occur in parallel without synchronization, yielding a high-throughput, statically scheduled scale-up solution. We support multiple scale-up factors by constructing multiple interconnects where each corresponds to different scale factors and select the desired factor at run-time as shown in Figure 5(c,d).

## 4.2 The MCA Architecture

In order to perform inference on high-definition images, the sampler architecture must support many labels per random variable. Unfortunately, the previously proposed SPA architecture and other prior work [40] limit the number of labels per random variable to 64. Because the magnitude of displacements and motion vectors scale with image resolution, this places a practical limit on the maximum input image that lies below HD resolution. We identify and resolve the issues limiting the scalability of the original proposal and articulate the improvements made to architect the MCA.

**4.2.1 Memory Overhead.** On-chip memory access (SRAM) is a significant part of the accelerator’s total power consumption, consuming up to 70% of the total power budget. With larger image inputs, the necessary magnitudes of motion vectors and displacements for optical flow and stereo-matching applications grow proportionally with the image size. As a result, the memory overheads for performing optical flow and stereo matching with MCMC grow super-linearly with the size of the inputs. Therefore, we explore how changing the shape of the topology impacts the energy footprint.

Configuring a MCA as a  $Q \times R$  grid of SPEs divides the input images evenly (i.e., a  $H \times W$  input image will be divided into  $\frac{H}{Q} \times \frac{W}{R}$  grid partitions and assigned to the corresponding SPE in the SPE grid). For each pixel in this grid, the MCMC process attempts to learn the correspondence between pixels stored in IMG1 and in IMG2. IMG1 stores pixels in a 1-to-1 correspondence with the assigned grid of pixels, while IMG2 must store *any* pixel that may be in correspondence with a pixel in IMG1. The IMG2 memory bank is the largest memory bank in an SPE by a significant margin. It can require hugely different amounts of memory depending on the dimensions of the

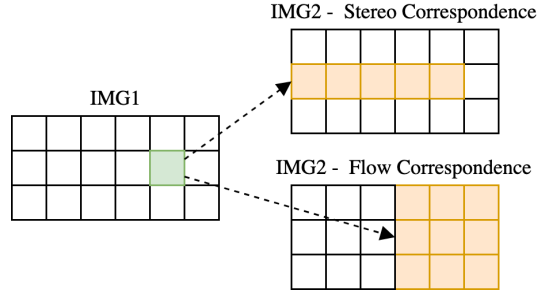


Fig. 7. Storing a single pixel in IMG1 (green), requires storing every pixel it can correspond to in IMG2 (orange).

SPE grid, image size, and the number of labels. To understand how these factors affect the memory overheads, we examine the overheads of the stereo matching, optical flow, and blind source separation applications more closely. Figure 7 shows one pixel in IMG1 and the possible correspondences to pixels in IMG2.

**Stereo Matching Memory Consumption** Stereo matching computes a disparity map between two images, which encodes the horizontal difference in coordinates between corresponding image points. If the accelerator supports labels that are  $L$  bits wide, the greatest unsigned horizontal difference it can represent is  $2^L - 1$ . Therefore, if we store a grid of  $N \times M$  pixels in IMG1 memory, we require space for  $N \times (M + 2^L - 1)$  pixels in IMG2 memory.

**Optical Flow Memory Consumption** Optical flow computes the motion vector for each point in an image between video frames. Each vector has a horizontal and vertical component. The upper bit range encodes the horizontal component, and the lower bit range encodes the vertical component. The radius of this vector, given that our label is  $L$  bits wide is  $r = 2^{\frac{L}{2}-1} - 1$ . Therefore, our IMG2 memory bank requires space for at least  $(N + 2r) \times (M + 2r)$  pixels.

The memory overhead of stereo matching is minimized when  $M \gg N$ , whereas the memory overhead for optical flow is minimized when  $M = N$ . We can control the magnitude of  $N$  and  $M$  by modifying the configuration of the SPE grid topology. For a  $Q \times R$  SPE topology,  $N = \frac{H}{Q}$ ,  $M = \frac{W}{R}$ , and so increasing the number of rows in the topology has an inverse effect on  $N$ . While intuition tells us that the optimal topology configuration is taller than it is wide ( $Q > R$ ), analytically solving for the optimal configuration is difficult because of the discontinuous search space and the non-linearity between memory size and power consumption. To find the lowest-power configuration, we perform a design space exploration in Section 5 that considers these insights.

**4.2.2 Work Distribution.** The MCA supports a maximum image size and may perform inference on any smaller resolution as well. While dynamic work scheduling is possible, this, again, breaks the symmetry that makes the MCA data movement so efficient. Instead, we pad the input to the maximum size and accept the work imbalance. Any SPEs that are assigned to only perform inference on padded variables are clock-gated to reduce power consumption.

### 4.3 Modifications for Blind Source Separation

In Section 3.1, we discussed how mixed-resolution sampling effectively reduces the computational costs of blind source separation on high-resolution images. However, without architectural modifications, BSS cannot immediately leverage the SPU or BigLittleMCA to support the high precision needed for sampling the mixing matrix and noise random variables. This section discusses the BSS problem structure and the hardware modifications necessary to utilize the SPU and BigLittleMCA.

Image Resolution	720x1280
Frames/Second	30
Sampling Iterations/Frame	200
Burn-in Iterations/Frame	200
Labels/Random Variable - Stereo Vision	256
Labels/Random Variable - Optical Flow	225

Table 3. Performance constraints for real-time inference

The input to BSS is  $K$  observed images,  $y_k \in Y^K$ , each of size  $M \times N$ . Each observed image  $y_k$  is believed to be a linear combination of up to  $L$  source images,  $s_l \in S^L$ . For a mixing matrix,  $A^{K \times L}$ , and a noise vector  $\mathbf{v}$ , we model the observed images as a linear combination of source images:

$$Y^K = S^L \times A^{L \times K} + \mathbf{v}$$

The task is to infer  $A$ ,  $S$  and  $\mathbf{v}$  given  $Y$ . Each source image,  $s_l$ , is modeled as an  $M \times N$  Markov Random Field, identical to the SPU problem setup. An iteration of the sampling algorithm has three steps. First, each RV in the MRF is sampled. Like the other applications, the energy function considers the smoothness with respect to the image and the local MRF state. The smoothness function used in [36] is an edge-preserving prior not currently implemented in the SPU but can be approximated using currently supported functions, or the SPU implementation can be changed.

Second, for each  $a_{l,k} \in A$ , we estimate the optimal  $a_{l,k}$  to minimize the residual between the predicted mixed image,  $\hat{y}_k = a_k \times s$ , and the observed image,  $y_k$ . This amounts to a single draw from a normal distribution for each random variable in  $A$  but requires us to compute a residual sum over the whole image. The residual for each pixel can be computed using hardware adders and multipliers already present in the SPU. These residuals then need to be summed over the entire field. This can be done by summing each residual for the pixels on a tile with an SPE and then using the existing SPE-to-SPE network to reduce these intermediates to the global residual. When these long-integer residuals are finished accumulating, we need to convert them to floating point and perform a series of other floating-point operations to compute the mean and standard deviation of the distribution from which the next  $a_{l,k}$  is drawn.

Third, we sample the noise parameter for the next iteration using the result of another residual sum. This time, the residuals derive the  $\lambda$  parameter for a gamma distribution,  $\Gamma(\alpha, \lambda)$ , from which we draw a sample to compute our noise parameter. The  $\alpha$  parameter is fixed as a function of the size of the MRF and is extremely large. For large values of  $\alpha$ , the gamma distribution can be approximated to high precision using a draw from a normal distribution, providing a re-use opportunity for any additional hardware needed from step two.

For each of these steps, the large expense lies in the per-pixel costs (e.g., sampling each pixel for  $s$  or globally accumulated residuals), not the additional sampling processes for noise and mixing variables. We profiled a 32-threaded CPU execution at  $720 \times 1280$  HD resolution, and the per-pixel computations comprised 99.95% of the total runtime. Since the high-precision sampling operations do not fall on the critical path, a single floating-point coprocessor will be sufficient for supporting these operations, incurring a minor area overhead. Therefore, with the addition of necessary control logic modifications and a floating-point coprocessor, the BigLittleMCA architectures can support blind source separation.

## 5 Hardware Evaluation

In this section, we evaluate the performance of our sampling approach, perform a design space exploration for our accelerator configuration, and investigate the power, area, and timing characteristics of our chosen design.

### 5.1 Implementation and Design Space Exploration Methodology

We implement all logic components of the BigLittleMCA accelerator design using Chisel HDL [5]. We synthesize and route our design at 850 MHz using Synopsys Design Compiler and Cadence Innovus, respectively, against a 65nm commercial technology library and SRAM memory compiler from TSMC and ARM. To better compare with prior and contemporary work, we scale these results to a 16nm technology node.

$$DelayFactor = a_{d_3}V^3 + a_{d_2}V^2 + a_{d_1}V + a_{d_0} \quad (2)$$

$$PowerFactor = a_{p_2}V^2 + a_{p_1}V + a_{p_0} \quad (3)$$

$$AreaFactor = a_{a_0} \quad (4)$$

$$P_x = \frac{PowerFactor_x}{PowerFactor_y} P_y \quad (5)$$

Stillmaker et al. [55] propose a methodology for scaling power, area, and delay results of one technology node to another using polynomials of varying degrees, shown in Equations 2, 3, and 4. The constants are determined for each technology node and are used to compute a factor for that technology. The ratio of these factors for the original and desired technology determines how to scale the area, delay, and power results. Equation 5 shows an example of scaling power, and a similar process applies for delay and area.

We scale our results using the factors in their manuscript and achieve an estimated 3GHz clock rate on our target 16nm node. We collect average cell activity statistics from Synopsys VCS while running one of our test data sets through an SPE and use them to accurately estimate the dynamic power of our design.

Section 3.1 showed that mixed-resolution sampling produces independent samples 11.7% more efficiently than standard Gibbs sampling for stereo vision. To normalize this difference in efficiency, we compare accelerators that produce the same number of independent samples per frame. For BigLittleMCA designs, we configure the accelerators for the parameters in Table 3. For MCA designs that do not use mixed-resolution sampling (i.e., SPAs), we analyze them using the same parameters as the comparison BigLittleMCA design, except that they must sample 224 iterations per frame to account for their lower sampling efficiency.

The results of this exploration are shown in Figure 8.

### 5.2 Results

This paper has proposed two major modifications to how tiled MCMC accelerators are constructed: mixed-resolution sampling and MCA topology optimization to reduce on-chip memory usage. However, these effects contribute differently to the overall power savings based on the input image resolution and performance requirements. To isolate their contributions, we investigate the impact of topology optimization both with and without mixed-resolution sampling by performing design space explorations (DSEs) over all feasible designs that meet the performance specifications in Table 3.

First, we investigate the impact of MCA topology optimization for the Big MCA of our BigLittle designs. This design space exploration is shown in Figure 8. The most efficient MCA design is a  $60 \times 8$  topology, which operates at 14.28W and uses 4.61MB of on-chip memory. This topology splits the MRF into  $12 \times 160$  slices for each SPE. We will use this  $60 \times 8$  configuration as the Big MCA for our BigLittle design. For comparison, a naive topology that splits the MRF into squares operates at 22.15W and uses 8.67MB of on-chip memory. Therefore, our optimization provides a 35.5% decrease in power usage while providing the same compute throughput and using 46.8% less on-chip memory.

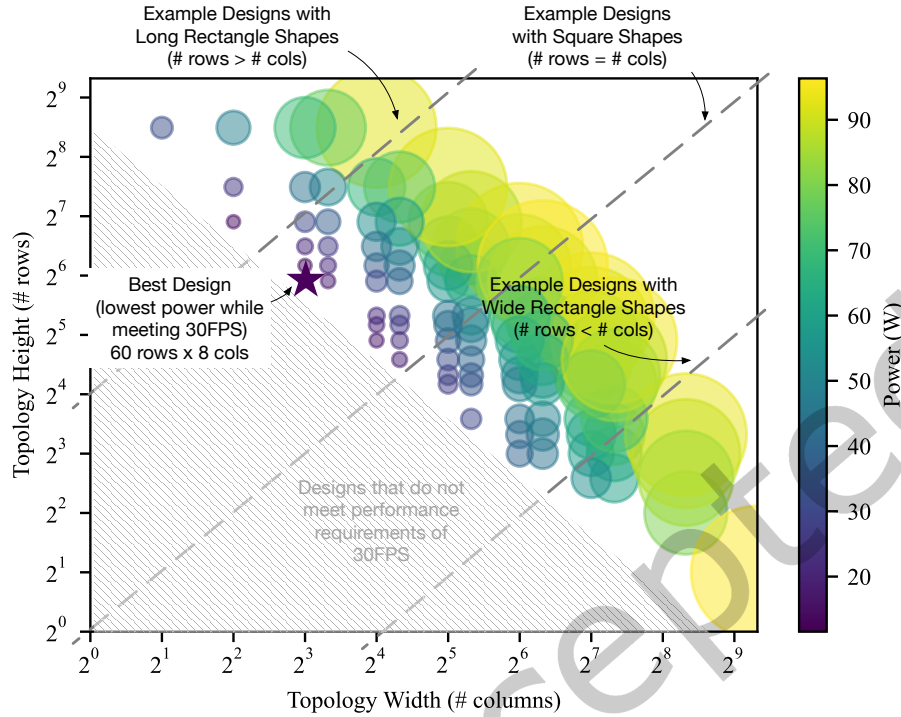


Fig. 8. This figure depicts the design space exploration for optimal MCA topology configurations given real-time HD performance constraints. For each design explored, the area and the color of the circle correspond to the power consumption; a bigger circle and a lighter color represent a higher power consumption. The diagonal dotted lines show topology shapes. The grayed-out area contains designs that do not meet the 30 FPS performance requirement. The lowest power design that meets the performance constraint (denoted by a star) is a long rectangle-shaped design with 60 rows and eight columns of SPEs.

And second, we investigate the impact of using the mixed-resolution approach. The resulting  $72 \times 16$  topology consumes 30.7W. These results show that using an ideal topology choice reduces power by 5.5% and uses 55% less SRAM than square tiling.

### 5.3 Hardware Implementation

Our prior analysis shows that a BigLittleMCA design using the aforementioned  $60 \times 8$  topology for the Big MCA and a  $30 \times 4$  topology for the Little MCA with a  $10\times$  and  $2\times$  scaling interconnect consumes 16.76W. We place and route a  $4\times$  scaled-down version of the BigLittleMCA design to estimate the power and area overheads of the distribution trees between the big and little MCAs in the mixed-resolution design. The scaled-down accelerator uses  $53.17\text{mm}^2$  of chip area and consumes 11.8W at 65nm. Using technology scaling factors [55] to scale to a 16nm design at 3GHz, we estimate that the full, non-scaled BigLittleMCA accelerator consumes  $26.9\text{mm}^2$  of area and 14.8W at a 16nm node, which aligns with our DSE results. The lower power of this design compared to our DSE's predicted power is largely due to the conservative SRAM power estimates reported by the SRAM compiler. In reality, the input/output pin toggle rate is lower than the 50% used to compute these estimates, leading to an

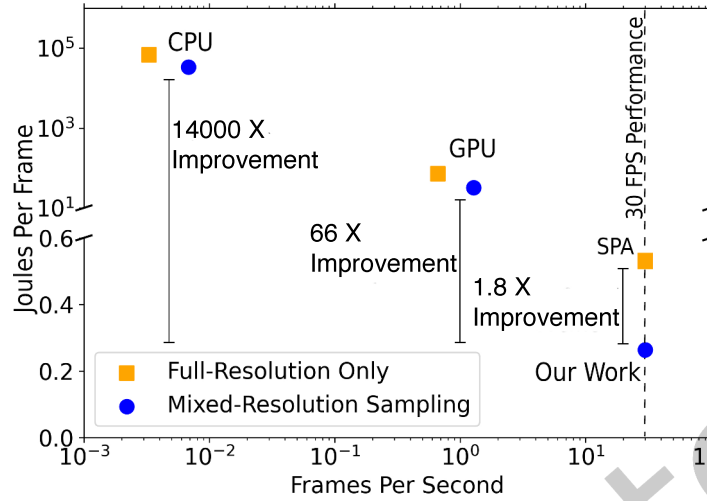


Fig. 9. Joules/Frame improvement of our work over the SPA [10] and software implementations of mixed-resolution sampling.

overestimation of SRAM power expenditure. Figure 9 shows the improvement of BigLittleMCA over prior work: energy per frame improvements of 14000 $\times$  for CPU and 66 $\times$  for GPU.

The numbers reported are conservative estimates that use register activity coefficients computed over a period when the accelerator is sampling from a random initial state. This is the highest-power phase of the algorithm because the large FIFO queues have high switching activity as the sequences they store are not converged. We use the switching activity data from this initial phase for our power analysis as a conservative estimate for the switching activity for the rest of the execution.

As the accelerator tiling grows, the practicalities of very-large-scale integration become more of a concern. While each SPEs and the MCAs are relatively unaffected due to their highly local communication grid, the Little-to-Big interconnect will become more challenging to route. We can consider that the amount of horizontal routing resources per row of SPE tiles is roughly constant and that all SPEs on the row must share these resources. As the architecture grows, so will the number of SPEs in a single row, resulting in increasing congestion. To combat this congestion, we routed the SPEs on lower metal layers to provide more routing resources for the interconnect.

The current interconnect implementation allows each of the hundreds of SPEs to operate in lockstep without explicit control, leading to an efficient SPE implementation. A general-purpose NoC will allow SPEs to share the data buses with the BigMCA and thereby reduce routing congestion at the cost of needing global synchronization to manage these buses.

#### 5.4 Alternative Input Resolutions

The necessity of extremely high resolution in computer vision inference depends on the specific use case. For instance, whereas 4K resolution might be superfluous for most applications of stereo vision, astrophysical imaging, which makes use of blind source separation [37], often does operate with extremely high-resolution imagery. At the same time, whereas stereo vision finds real-time use in robotics, blind source separation does not have a similar need for real-time performance.

The sizes of the BigLittleMCA tilings can be scaled up or down to support larger or smaller input image dimensions. To estimate how our design would behave for different problem sizes, we perform the same design

Max Res.	# Pixels	BigMCA Dims.	Power (% imp.)	On-Chip Memory
$300 \times 300$	90K	$30 \times 2$	2.1W (27.5%)	470 KB
720p	921K	$60 \times 8$	16.76W (48.5%)	4.61 MB
1080p	2M	$180 \times 12$	78.01W (48.2%)	17.57 MB
4K	8.29M	$540 \times 32$	631.73W (49.0%)	157.8 MB

Table 4. Power consumption and percent improvement for accelerators for a range of image resolutions. Each BigLittleMCA dimension provides just the size of the BigMCA, and the LittleMCA is a 2× scaled down version.

space exploration for an edge design point ( $300 \times 300$  images), 720p ( $720 \times 1280$ ), 1080p ( $1080 \times 1920$ ), and 4K ( $2160 \times 3840$ ). We scale the number of labels used for each resolution so that the maximum magnitude vector for stereo matching and optical flow is proportionately the same as using 256 labels for 720p resolution. Table 4 shows the results of this exploration.

We observe that at lower resolutions, our improvements are smaller than those of prior work. This is because, as the resolution shrinks, so does the number of required labels, thereby reducing the effect to which topology optimization reduces data duplication across SPEs. As the images grow larger, the benefit is made more extreme, increasing the savings to nearly 50%.

CoopMC [11] is a recent MCMC work that computes probabilities for all random variable labels in parallel using separate computation pipelines. The probabilities are then merged using a tree structure to produce a random variable sample with a lower time complexity,  $O(\log N)$ , compared to  $O(N)$  for traditional sequential samplers, for  $N$  labels per random variable. CoopMC has a higher area-per-sampler but achieves better performance-per-area and resource utilization due to their  $O(\log N)$  runtime. However, the cost of this approach is a relatively low 500MHz clock rate using a 12nm technology node and 16-bit datatype in comparison to our achieved clock rate of 850MHz using a 65nm technology node and the SPE’s 8-bit datatype.

To compare CoopMC against the BigLittleMCA approach, we estimate the cost of a CoopMC-based tiled accelerator. We use the authors’ 64-label end-to-end case study for the power and sampling performance of an individual sampler, 9.53mW and 1.53× improvement over sequential sampling, respectively. We assume the same on-chip memory architecture as the SPE for a CoopMC tile. We synthesize a BigLittleMCA Accelerator at 500MHz as a point of comparison against CoopMC at 500MHz. We perform the same design space search from Section 5 for SPA, BigLittleMCA, and CoopMC and report these results in Table 5 for different combinations of number of labels and clock rates of the various architectures. Each resulting implementation still achieves the real-time inference constraints from Table 3.

We find that a 64-label BigLittleMCA running at 500MHz consumes 4.49W of power and 21.26mm<sup>2</sup> of chip area, whereas a CoopMC tiled accelerator consumes 44.21W and 45.58mm<sup>2</sup> of chip area. If we use the CoopMC sampler inside a BigLittle tiling, it consumes less power and area: 25.67W and 25.33mm<sup>2</sup>.

Interestingly, when we only need 64 labels for inference, a 500MHz BigLittleMCA accelerator consumes 9.8% less power than BigLittleMCA at 3GHz. The cost of this, however, is that we need 6× the number of units and 4.3× more chip area to achieve real-time performance. This relationship is not mirrored for the 256-label BigLittleMCA accelerators because SRAM area and power overheads are more significant for a higher number of labels. These results demonstrate that BigLittleMCA achieves state-of-the-art area and power results over prior work.

## 6 Related Work

**Deep-Learning Accelerators** The EdgeTPU [4] is a recent variation of Google’s popular Tensor Processing Unit (TPU) [34] that targets mobile and edge environments. The EdgeTPU operates at 2W and is able to execute a variety of popular mobile deep-learning CV models at or exceeding real-time speeds [22]. Recent convolutional neural

	BigLittleMCA		SPA	BigLittleMCA		BigLittle CoopMC	SPA CoopMC
	3 GHz	500 MHz	3 GHz	3 GHz	500 MHz	500 MHz	500 MHz
Clock Rate	3 GHz	500 MHz	3 GHz	3 GHz	500 MHz	500 MHz	500 MHz
Labels Accuracy	256 High	256 High	256 High	64 Medium	64 Medium	64 Medium	64 Medium
Topology Dims	Big: 60x8 Little: 30x2	Big: 180x16 Little: 90x8	72x16	Big: 30x4 Little: 15x2	Big: 72x10 Little: 36x5	Big: 60x8 Little: 30x4	8x128
Area	16.31mm <sup>2</sup>	78.86mm <sup>2</sup>	27.58mm <sup>2</sup>	6.46mm <sup>2</sup>	21.26mm <sup>2</sup>	25.3mm <sup>2</sup>	45.58mm <sup>2</sup>
Power	16.76W	17.4W	30.7W	4.98W	4.49W	25.67W	44.21W

Table 5. Comparison of BigLittleMCA against prior work [10, 11]. Whereas the area and power of the SPE sampler do not scale significantly as a function of the number of labels, CoopMC [11] achieves higher sampler performance by increasing the compute quadratically with the number of labels. We compare our approach against their 64-label CoopMC sampler. While using fewer labels is an acceptable approach for stereo vision [52, 53], it can severely impact inference accuracy for other tasks, namely optical flow and BSS, which require more labels to account for the problem’s higher degree of freedom and to ensure convergence, respectively.

network accelerator works such as Eyeriss v2 [12] take advantage of network sparsity and weight quantization to increase power efficiency, operating at approximately 0.5W. The mobile networks these accelerators target have input sizes of approximately  $300 \times 300$ . Downscaling high-resolution inputs to this input size will incur information loss, resulting in higher error rates. As a point of comparison, we estimate the power of a BigLittleMCA accelerator that performs real-time inference at a  $300 \times 300$  resolution in Table 4. While these neural network accelerators support a greater variety of applications through the flexibility of neural networks, our  $300 \times 300$  and 720p designs operate using similar power budgets and are not black box methods. MCMC algorithms provide interpretability because it is straightforward to see how the MCMC state and image inputs contribute to the inference. Further, they often provide statistical guarantees (e.g., convergence) and confidence metrics by measuring the variance of output samples. Recent work in interpretable deep learning approaches has yielded networks for tasks such as optical flow [27, 58] and object classification [9]. However, it is unclear how amenable these networks are to execution in real-time, edge environments. Therefore, our design is a viable option for image processing inference where interpretability and confidence measures are desirable.

**Vision Processing Units** Several commercial products have been developed explicitly to accelerate vision workloads [16, 29, 50, 56]. These architectures take various forms but usually utilize some combination of wide vector ALUs and neural network acceleration to provide high frame rate HD inference. For instance, the Intel Movidius Myriad 2 [16] can achieve 60FPS 720p stereo matching at 0.9W. However, the algorithms that these chips use are black-box models and do not, to our knowledge, support per-pixel inference confidence.

**MCMC Accelerators** Most existing MCMC Accelerators target either the architecture of a single functional unit [11, 40, 59] or an FPGA design for accelerating large inference problems [10, 41, 46]. The FPGA designs do not scale to HD images due to the shortage of on-chip memory. We show in Section 4.2 how appropriate memory tiling can reduce the on-chip memory requirements by 46.8% compared to prior work, increasing the scalability of existing FPGA designs. Even with these improvements, the resource limitations of FPGAs prevent practical MCMC sampling for HD image inference.

CoopMC [11] is a recent work in accelerating MCMC that proposes a sampler with lower time complexity ( $\mathcal{O}(\log N)$  for  $N$  labels per RV) than prior samplers at the cost of area. We note that CoopMC sampler units could be used in place of SPUs whenever there is sufficient chip area to support them. CoopMC was designed to be a "plug-and-play" replacement for samplers like FlexGibbs [40] and the SPU [59]. However, the area of their

sampler scales exponentially with the number of labels, and as a result, certain applications with small label spaces may benefit from CoopMC samplers.

**Pre/Post-Processing** While stereo vision and motion estimation are inference tasks, they can also be used as a preprocessing step for other computer vision models. For instance, a common approach for gesture recognition inference is to take in stereo vision and/or motion estimation data and image inputs [2, 48, 49]. Separate hardware accelerators for preprocessing have been proposed in previous work [23] for operations like image resizing, normalization, and rotation. Like those methods, a key benefit of BigLittleMCA as a preprocessor is that all operands are stored on-chip — there are no weights to store. As a result, integration alongside existing neural network accelerators (e.g., EdgeTPU) would provide additional input modalities for ML models without any need for communication with off-chip memory, the main bottleneck for neural network models. Such a fusion of CNN (e.g., GPUs) and MCMC accelerators to collaborate on a task has been proposed previously by Liu et al. [47]. In their work, the CNN performs object detection and passes those results to the MCMC accelerator as the input to an MCMC-based pose estimation algorithm.

## 7 Conclusion

In this paper, we propose mixed-resolution sampling for computer vision applications with smoothness priors. Our approach reduces the computational resources of inference while increasing accuracy by 18%, 13.8%, and 6.3% for stereo matching, optical flow, and blind source separation tasks, respectively.

We design a BigLittleMCA accelerator that supports more labels than prior work per random variable and enables real-time 720p HD inference. We investigate how the topology configuration of the architecture affects memory usage, find significant inefficiencies in prior designs, and reduce the on-chip memory usage by 46.8%. We synthesize an ASIC BigLittleMCA design for 720p real-time inference using 256 labels per random variable and show our design reduces the area footprint and power consumption by 47.8% and 48.5% relative to prior work.

## Acknowledgments

This project is supported in part by Intel, in part by the Semiconductor Research Corporation, in part by a National Science Foundation award CNS-1616947, and in part by a National Science Foundation CAREER award CCF-2045973.

## References

- [1] 2013. big.LITTLE Technology: The Future of Mobile. <https://armkeil.blob.core.windows.net/developer/Files/pdf/white-paper/big-little-technology-the-future-of-mobile.pdf>.
- [2] Mahdi Abavisani, Hamid Reza Vaezi Joze, and Vishal M. Patel. 2019. Improving the Performance of Unimodal Dynamic Hand-Gesture Recognition With Multimodal Training. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 1165–1174. doi:10.1109/CVPR.2019.00126
- [3] Jesse Adams, Matthias Morzfeld, Kevin Joyce, Marylesa Howard, and Aaron Luttman. 2021. A blocking scheme for dimension-robust Gibbs sampling in large-scale image deblurring. *Inverse Problems in Science and Engineering* 29, 12 (2021), 1789–1810. doi:10.1080/17415977.2021.1880398 arXiv:<https://doi.org/10.1080/17415977.2021.1880398>
- [4] Alphabet. 2023. Edge TPU. <https://cloud.google.com/edge-tpu>.
- [5] Jonathan Bachrach, Huy Vo, Brian Richards, Yunsup Lee, Andrew Waterman, Rimas Avizienis, John Wawrzynek, and Krste Asanović. 2012. Chisel: Constructing Hardware in a Scala Embedded Language. In *Proceedings of the 49th Annual Design Automation Conference (San Francisco, California) (DAC '12)*. Association for Computing Machinery, New York, NY, USA, 1216–1225. doi:10.1145/2228360.2228584
- [6] Simon Baker, Stefan Roth, Daniel Scharstein, Michael J. Black, J.P. Lewis, and Richard Szeliski. 2007. A Database and Evaluation Methodology for Optical Flow. In *2007 IEEE 11th International Conference on Computer Vision*. 1–8. doi:10.1109/ICCV.2007.4408903
- [7] Subho S. Banerjee, Zbigniew T. Kalbarczyk, and Ravishankar K. Iyer. 2019. AcMC 2 : Accelerating Markov Chain Monte Carlo Algorithms for Probabilistic Models. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (Providence, RI, USA) (ASPLOS '19)*. Association for Computing Machinery, New York, NY, USA, 515–528. doi:10.1145/3297858.3304019

- [8] Stephen T. Barnard. 1989. Stochastic stereo matching over scale. *International Journal of Computer Vision* 3, 1 (1989), 17–32. doi:10.1007/BF00054836
- [9] Alina Jade Barnett, Fides Regina Schwartz, Chaofan Tao, Chaofan Chen, Yin hao Ren, Joseph Y Lo, and Cynthia Rudin. 2021. A case-based interpretable deep learning model for classification of mass lesions in digital mammography. *Nature Machine Intelligence* 3, 12 (2021), 1061–1070. doi:10.1038/s42256-021-00423-x
- [10] Ramin Bashizade, Xiangyu Zhang, Sayan Mukherjee, and Alvin R Lebeck. 2021. Accelerating Markov Random Field Inference with Uncertainty Quantification. *arXiv preprint arXiv:2108.00570* (2021).
- [11] Yuji Chai, Glenn G. Ko, Wei-Te Mark Ting, Luke Bailey, David Brooks, and Gu-Yeon Wei. 2022. CoopMC: Algorithm-Architecture Co-Optimization for Markov Chain Monte Carlo Accelerators. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 38–52. doi:10.1109/HPCA53966.2022.00012
- [12] Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, and Vivienne Sze. 2019. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 2 (2019), 292–308.
- [13] Ting-Wu Chin, Ruizhou Ding, and Diana Marculescu. 2019. AdaScale: Towards Real-time Video Object Detection using Adaptive Scaling. In *Proceedings of Machine Learning and Systems*, A. Talwalkar, V. Smith, and M. Zaharia (Eds.), Vol. 1. 431–441. <https://proceedings.mlsys.org/paper/2019/file/b1d10e7bafa4421218a51b1e1f1b0ba2-Paper.pdf>
- [14] Eunah Choi, Sangyoon Lee, and Hyunki Hong. 2017. Hierarchical Stereo Matching in Two-Scale Space for Cyber-Physical System. *Sensors (Basel, Switzerland)* 17, 7 (07 2017), 1680. doi:10.3390/s17071680
- [15] Gregory F. Cooper. 1990. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence* 42, 2 (1990), 393–405. doi:10.1016/0004-3702(90)90060-D
- [16] Intel Corporation. 2022. Intel® Movidius™ Myriad™ X Vision Processing Unit (VPU) with Neural Compute Engine. <https://intel.ly/3jiGwpl>.
- [17] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. 2017. Deformable Convolutional Networks. doi:10.48550/ARXIV.1703.06211
- [18] Javier Felip, Nilesh Ahuja, and Omesh Tickoo. 2019. Tree pyramidal adaptive importance sampling. doi:10.48550/ARXIV.1912.08434
- [19] Yihan Fu, Daijing Shi, Anjunyi Fan, Wenshuo Yue, Yuchao Yang, Ru Huang, and Bonan Yan. 2024. Probabilistic Compute-in-Memory Design for Efficient Markov Chain Monte Carlo Sampling. *IEEE Transactions on Circuits and Systems I: Regular Papers* 71, 2 (2024), 703–716. doi:10.1109/TCSI.2023.3337529
- [20] Andrew Gelman and Donald B. Rubin. 1992. Inference from Iterative Simulation Using Multiple Sequences. *Statist. Sci.* 7, 4 (Nov. 1992), 457–472. doi:10.1214/ss/1177011136 Publisher: Institute of Mathematical Statistics.
- [21] Ross Girshick. 2015. Fast R-CNN. doi:10.48550/ARXIV.1504.08083
- [22] Google. 2022. <https://coral.ai/docs/edgetpu/benchmarks/>.
- [23] Dongho Ha, Won Woo Ro, and Hung-Wei Tseng. 2023. TensorCV: Accelerating Inference-Adjacent Computation Using Tensor Processors. In *2023 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. 1–6. doi:10.1109/ISLPED58423.2023.10244461
- [24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2014. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In *Computer Vision – ECCV 2014*. Springer International Publishing, 346–361. doi:10.1007/978-3-319-10578-9\_23
- [25] Yinlin Hu, Rui Song, and Yunsong Li. 2016. Efficient coarse-to-fine patchmatch for large displacement optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5704–5712.
- [26] Yaoyu Hu, Wenshan Wang, Huai Yu, Weikun Zhen, and Sebastian Scherer. 2021. ORStereo: Occlusion-Aware Recurrent Stereo Matching for 4K-Resolution Images. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 5671–5678.
- [27] Bowen Huang, Jinjia Zhou, Xiao Yan, Ming’e Jing, Rentao Wan, and Yibo Fan. 2021. CS-MCNet: A Video Compressive Sensing Reconstruction Network with Interpretable Motion Compensation. In *Computer Vision – ACCV 2020*, Hiroshi Ishikawa, Cheng-Lin Liu, Tomas Pajdla, and Jianbo Shi (Eds.). Springer International Publishing, Cham, 54–67.
- [28] Tak-Wai Hui, Xiaou Tang, and Chen Change Loy. 2018. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8981–8989.
- [29] Mircea Horea Ionica and David Gregg. 2015. The Movidius Myriad Architecture’s Potential for Scientific Computing. *IEEE Micro* 35, 1 (2015), 6–14. doi:10.1109/MM.2015.4
- [30] Claus S. Jensen, Uffe Kjærulff, and Augustine Kong. 1995. Blocking Gibbs Sampling in Very Large Probabilistic Expert Systems. *Int. J. Hum.-Comput. Stud.* 42, 6 (jun 1995), 647–666. doi:10.1006/ijhc.1995.1029
- [31] Claus Skaanning Jensen and Augustine Kong. 1999. Blocking Gibbs Sampling for Linkage Analysis in Large Pedigrees with Many Loops. *The American Journal of Human Genetics* 65, 3 (1999), 885–901. doi:10.1086/302524
- [32] Natalie Enright Jerger, Tushar Krishna, and Li-Shiuan Peh. 2017. On-Chip Networks: Second Edition. *Morgan and Claypool Publishers* (2017).
- [33] Fang Jian, Zheng Wei, Zhang Ding, and Wang Kuang. 2007. A mode correlation-based fractional pixel motion estimation for H.264 video coding. In *2007 7th International Conference on ASIC*. 938–941. doi:10.1109/ICASIC.2007.4415786

- [34] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. 2017. In-Datcenter Performance Analysis of a Tensor Processing Unit. doi:10.48550/ARXIV.1704.04760
- [35] Kiran Kale, Sushant Pawar, and Pravin Dhulekar. 2015. Moving object tracking using optical flow and motion vector estimation. In *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)*. 1–6. doi:10.1109/ICRITO.2015.7359323
- [36] Koray Kayabol, Ercan E. Kuruoglu, and Bülent Sankur. 2009. Bayesian Separation of Images Modeled With MRFs Using MCMC. *IEEE Transactions on Image Processing* 18, 5 (2009), 982–994. doi:10.1109/TIP.2009.2012905
- [37] Koray Kayabol, Ercan E. Kuruoglu, Bulent Sankur, Emanuele Salerno, and Luigi Bedini. 2009. Fast MCMC separation for MRF modelled astrophysical components. In *2009 16th IEEE International Conference on Image Processing (ICIP)*. 2769–2772. doi:10.1109/ICIP.2009.5414190
- [38] Soroosh Khoram, Kyle Daruwalla, and Mikko Lipasti. 2023. Energy-Efficient Bayesian Inference Using Bitstream Computing. *IEEE Computer Architecture Letters* 22, 1 (2023), 37–40. doi:10.1109/LCA.2023.3238584
- [39] Genshiro Kitagawa and Will Gersch. 1996. *Smoothness priors analysis of time series*. Vol. 116. Springer Science & Business Media.
- [40] G. G. Ko, Y. Chai, R. A. Rutenbar, D. Brooks, and G. Wei. 2019. FlexGibbs: Reconfigurable Parallel Gibbs Sampling Accelerator for Structured Graphs. In *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 334–334. doi:10.1109/FCCM.2019.00075 ISSN: 2576-2621.
- [41] Glenn G. Ko, Yuji Chai, Rob A. Rutenbar, David Brooks, and Gu-Yeon Wei. 2019. Accelerating Bayesian Inference on Structured Graphs Using Parallel Gibbs Sampling. In *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*. 159–165. doi:10.1109/FPL.2019.00033
- [42] J. Konrad and E. Dubois. 1992. Bayesian estimation of motion vector fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14, 9 (1992), 910–927. doi:10.1109/34.161350
- [43] John K. Kruschke. 2015. Chapter 7 - Markov Chain Monte Carlo. In *Doing Bayesian Data Analysis (Second Edition)* (second edition ed.), John K. Kruschke (Ed.). Academic Press, Boston, 143–191. doi:10.1016/B978-0-12-405888-0.00007-6
- [44] Junggi Lee, Kyeongbo Kong, Gyu-jin Bae, and Woo-Jin Song. 2020. BlockNet: A Deep Neural Network for Block-Based Motion Estimation Using Representative Matching. *Symmetry* 12, 5 (2020). doi:10.3390/sym12050840
- [45] Jun S. Liu. 2001. Monte Carlo Strategies in Scientific Computing. *Springer Publishing Company, Incorporated* (2001).
- [46] Shuanglong Liu, Grigorios Mingas, and Christos-Savvas Bouganis. 2017. An Unbiased MCMC FPGA-Based Accelerator in the Land of Custom Precision Arithmetic. *IEEE Trans. Comput.* 66, 5 (2017), 745–758. doi:10.1109/TC.2016.2630682
- [47] Yanqi Liu, Giuseppe Calderoni, and Ruth Iris Bahar. 2020. Hardware Acceleration of Monte-Carlo Sampling for Energy Efficient Robust Robot Manipulation. In *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*. 284–290. doi:10.1109/FPL50879.2020.00054
- [48] Hasan Mahmud, Mashrur M Morshed, and Md Kamrul Hasan. 2021. A deep learning-based multimodal depth-aware dynamic hand gesture recognition system. *arXiv preprint arXiv:2107.02543* (2021).
- [49] Pavlo Molchanov, Xiaodong Yang, Shalini Gupta, Kihwan Kim, Stephen Tyree, and Jan Kautz. 2016. Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3D Convolutional Neural Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4207–4215. doi:10.1109/CVPR.2016.456
- [50] David Moloney, Brendan Barry, Richard Richmond, Fergal Connor, Cormac Brick, and David Donohoe. 2014. Myriad 2: Eye of the computational vision storm. In *2014 IEEE Hot Chips 26 Symposium (HCS)*. 1–18. doi:10.1109/HOTCHIPS.2014.7478823
- [51] Jingfu Ren and Wang Hanbo. 2018. Application of Stereo Vision Technology In 3D Reconstruction of Traffic Objects. In *2018 17th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*. 100–102. doi:10.1109/DCABES.2018.00035
- [52] Daniel Scharstein and Chris Pal. 2007. Learning Conditional Random Fields for Stereo. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 1–8. doi:10.1109/CVPR.2007.383191
- [53] D. Scharstein, R. Szeliski, and R. Zabih. 2001. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*. 131–140. doi:10.1109/SMBV.2001.988771
- [54] Robert J Shiller. 1984. Smoothness priors and nonlinear regression. *J. Amer. Statist. Assoc.* 79, 387 (1984), 609–615.
- [55] Aaron Stillmaker and Bevan Baas. 2017. Scaling equations for the accurate prediction of CMOS device performance from 180nm to 7nm. *Integration* 58 (2017), 74–81. doi:10.1016/j.vlsi.2017.02.002

- [56] Elene Terry. 2019. Silicon at the Heart of HoloLens 2. In *2019 IEEE Hot Chips 31 Symposium (HCS)*. 1–26. doi:10.1109/HOTCHIPS.2019.8875669
- [57] Siyang Wang, Xiangyu Zhang, Yuxuan Li, Ramin Bashizade, Song Yang, Chris Dwyer, and Alvin R. Lebeck. 2016. Accelerating Markov Random Field Inference Using Molecular Optical Gibbs Sampling Units. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. 558–569. doi:10.1109/ISCA.2016.55
- [58] Jian Zhang and Bernard Ghanem. 2018. ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1828–1837.
- [59] Xiangyu Zhang, Ramin Bashizade, Craig LaBoda, Chris Dwyer, and Alvin R. Lebeck. 2018. Architecting a Stochastic Computing Unit with Molecular Optical Devices. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, Los Angeles, CA, 301–314. doi:10.1109/ISCA.2018.00034
- [60] Xiangyu Zhang, Ramin Bashizade, Yicheng Wang, Sayan Mukherjee, and Alvin R. Lebeck. 2021. Statistical Robustness of Markov Chain Monte Carlo Accelerators. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (Virtual, USA) (ASPLOS 2021)*. Association for Computing Machinery, New York, NY, USA, 959–974. doi:10.1145/3445814.3446697

Received 27 July 2024; revised 8 March 2025; accepted 11 April 2025

Just Accepted